JSiSE Research Report
vol.33,no.1(2018-5)

オブジェクト指向プログラミングの利便性に着目した学習手法の改善と評価

竹川 夏実*，仲林 清**
*千葉工業大学大学院，**千葉工業大学

Improvement and Evaluation of a Learning Method for Object-oriented Programming Focusing on its Advantages

Natsumi Takekawa*，Kiyoshi Nakabayashi**
*Graduate School of Chiba Institute of Technology **Chiba Institute of Technology

オブジェクト指向プログラミング（以下 OOP）はプログラムの機能拡張に優れる手法だが，苦手意識を持つ学習者が多いとされている．OOP を正しく利用するには，まず OOP の利便性を理解した上で，基礎概念と利便性の結びつきを意識することが有効ではないかと考えられる．そこで，前回の報告では，OOP と非 OOP との比較学習を提案し，OOP の利便性を認識させることを狙った．今回の報告では，さらに学習者の理解度を深めるため，OOP と非 OOP の比較学習を支援するような改善策を提案した．具体的には，流れ図課題や改善した事後問題を用意し，これらを比較学習と併用した際の効果を検証した．

キーワード: オブジェクト指向プログラミング，手続き型との比較学習，利便性の理解，流れ図

1. はじめに

オブジェクト指向プログラミング(以下 OOP)は,機能拡張を前提とするプログラムを開発することに優れた手法である(1)．OOP を正しく利用するには、「クラス」,「継承」,「多態性」などの OOP の基礎概念を理解する必要がある.しかし，この基礎概念が抽象的で理解が困難であるため，OOP そのものに苦手意識を持つ学習者が多数存在する．

この問題を解決するための指導方法には，OOP の基礎概念を現実社会の物体に投影させる手法(2)，プログラムの動きを可視化する手法(3-5),などがある．これらの手法では，OOP の基礎概念の振る舞いを理解させることはできるが，基礎概念の必要性やそれによって生じる利点を理解させることは困難である．

そこで本研究では，OOP の利便性である「拡張性」，「保守性」を実感させ，これらが OOP の基礎概念によって生じていることを理解させる，というアプローチをとる．学習者の OOP の基礎概念に関する理解度を人に説明ができるレベルまで引き上げることを目的とする．また，OOP の利便性を実感させるため，OOP と手続き型で同じ課題を解かせ,プログラミングの手間や複雑さを比較させる．今回の報告では，前回の報告(7)を踏まえ，学習者の OOP に関する理解度をさらに向上させるためのアプローチを提案する．そして，そのアプローチを，OOP と手続き型との比較学習に併用した際の効果について調査する．以下，第 2 章では本研究の目的について述べ，第 3 章では，学習目標について述べる．第 4 章では学習目標を達成するための学習手法について説明する，第 5 章で実験結果を，第 6 章で考察を行い，第 7 章で今後の課題を述べる．

2. 前回の課題と今回の目的

2.1 前回までの学習目標

本研究では学習者の OOP の理解度向上を狙う．それに伴い，OOP の「拡張性」,「保守性」といった利便性を実感させ，その利便性がクラスや継承，多態性などの OOP の基礎概念によって生じることを理解させることを目的とする．これを達成するため，前回の報告(6)では，OOP と非 OOP（手続き型プログラミング）による比較学習を提案し，その効果を調査した．具体的には，OOP と非 OOP のサンプルプログラムを用意し，学習者に両手法で拡張のしやすさの違いを考えさせる．この時，学習者には非 OOP にはない OOP の利便性を実感させ，この利便性が生じる理由として，クラスや継承，多態性といった OOP の基礎概念がプログラム内で作用していることを理解させる．

2.2 前回の実験結果と考察

前回の実験では，比較学習により，ほとんどの学習者が高い自己評価を出していた．特に拡張性，保守性といった OOP の利便性を実感する手助けになっていることがわかった．このことから，比較学習は OOP の理解度向上に役立つといえる．一方で，利便性は実感できても，利便性が生じる理由を押さえるまでの理解ができた学習者は少なかった．原因は，OOP の利便性が OOP の基礎概念と結びついているということを十分に意識させることができなかったことにある．具体的には，課題中で学習者に OOP の利便性と基礎概念を意識させるタイミングを明確にしていなかったことが挙げられる．OOP の利便性と基礎概念を意識させるための学習手法であるのにも関わらず，課題で意識させていたのはクラスや多態性などの個々の基礎概念を理解することであった．そのため，「OOP の基礎概念の作用によって利便性が発生することを，学習者が理解できたか」という点に関して，学習者を明確に評価することができなかった．

2.3 今回の目的

2.2 節の考察を踏まえ，本研究では，学習者に「OOP の利便性が生じる理由が，OOP の基礎概念にある」ということを課題の中で意識させ，最終的に人に説明ができるレベルまでの理解度にさせることを狙う．これを達成するため，学習目標の改善として，評価基準の見直しを行う．また，学習手法の改善として，比較学習に加え流れ図課題を追加する．さらに，新しい評価基準を達成するために事後問題の構成の見直しを行う．それぞれの具体的な内容については，評価基準は 3 章，流れ図課題と事後問題については 4 章で説明する．

3. 学習目標

前回の報告では，OOP の利便性と概念について，それぞれ独立した学習目標を設定していた．これに対し今回は，両者を関連付けて理解する学習目標に修正した．前回は，OOP の基礎概念に関して「クラス」，「継承」，「多態性」の 3 つ，利便性に関して「拡張性」，「保守性」の 2 つをそれぞれ学習目標とした．しかし，この学習目標を評価の基準にすると，それぞれの項目同士を結びつけた評価ができなくなる．そのため，学習者の OOP の基礎概念と利便性の結びつきの理解度を正しく判断できていなかった可能性がある．

そこで今回は，OOP の利便性が生じる理由と関連する基礎概念を結びつけた説明ができることを学習目標とし，評価基準を修正した．この評価基準を表 1 に示す．レベル 0 は，機能拡張の課題を全て完了したが，拡張性・保守性といった利便性を体感できない程度の理解度を示す．さらに，レベル 1 から 3 は，OOP の利便性を実感し（レベル 1），それが基礎概念と関連していることを具体例で（レベル 2），あるいは一般的に（レベル 3）説明ができるレベルである．レベル 4 は，OOP の利便性の特徴とそれが基礎概念のどのような作用から生じるかについて，基礎概念を列挙しながら説明できるレベルである．

表 1 評価基準

レベル	内容	備考
0	OOP の利便性が体感できない	課題を通しても OOP に利便性を感じられない
1	OOP の利便性が体感的にわかる	課題を通して，OOP に利便性があることが体感的に理解できる
2	具体例を用いて OOP の利便性と基礎概念を説明できる	プログラミング課題の拡張処理を例に挙げながら，利便性と基礎概念の関係性が説明できる
3	OOP の利便性について，具体例なしで一般的に説明できる	拡張性・保守性について，一般的に説明できる
4	OOP には２つの利便性があり，その利便性は基礎概念と結びついているということが説明できる．	利便性と基礎概念の関係性について，今まで学習してきたことをベースに自分の言葉で説明できる．

4. 学習手法

4.1 基本的な方針

本研究では，OOP の理解度向上のため，OOP と手続き型による比較学習を行う．プログラムの比較学習に加え，今回は，流れ図課題や内容・配置を改善した事後問題を用意し，比較学習と併用した際の効果について検証した．以下の節でそれぞれについて説明する．

4.2 比較学習

本研究では，学習者に手続き型と OOP でプログラムを比較させるという手法をとる．OOP の基礎概念を

利用しない非 OOP（手続き型）を比較対象にすることで，OOP の利便性を実感させやすくする．

課題のプログラムは OOP と手続き型の両手法で同一のデータ構造を取っているが，処理方法が異なっている．手続き型では，主に if 文の分岐処理でデータを分別し，計算や出力等の処理を行っていく．つまり，新しい種類のデータを追加していくほど分岐が増加し，ソースコードが複雑化していく．一方 OOP では，クラスを利用してデータの分別をし，計算・出力等の処理もそのクラス内で行うためソースコードが複雑化しない．これは，クラスをはじめとした継承・多態性などの OOP の基礎概念がプログラム中で利用されているからであり，その結果として「修正箇所が限定的になる」ことや，「バグが発生しても発見しやすくなる」という OOP の利便性が実現される．

4.3 学習課題

今回の実験で用意した学習課題は，プログラミング課題，流れ図課題，OOP の基礎概念と利便性の関係性に関する自由記述問題（事後問題）である．前回から変更した点は，流れ図課題の追加と事後問題の配置・内容の改善の２点である．それぞれの具体的な内容に関しては，次節から詳しく説明する．

4.3.1 プログラミング課題

作成したサンプルプログラムは，ショッピングカートの実装をテーマとしたものである．このプログラムは，顧客１人１人が買い物かごを取り，好きな商品を入れ，レジで精算するという一連の処理を行うものである．本研究で取り上げた「ショッピング」とは，amazon のようなショッピングサイトを想定し，さまざまな品種の商品が購入できるものと定義している．

このプログラミング課題では，商品種別を順次追加することで拡張を行っていく．プログラムの機能要件は，「商品種別ごとの割引率を変えること」であり，OOP の実装条件は「割引計算を商品種別ごとのクラスのメソッドのオーバーライドで実現させること」である．実験群では，OOP と手続き型でどのような差異がでるかを意識してもらう．

4.3.2 流れ図課題

今回の実験では，各プログラミング課題を実施したあとに流れ図に関する課題を実施している．プログラミング課題で拡張した箇所を，拡張前のサンプルプログラムに関する流れ図に追加してもらう課題である．流れ図課題を用意した意図は，拡張したプログラムの処理を描き込むことで，拡張の前後でどのような変化があったかを振り返らせることにある．手続き型プログラムに対応した流れ図も用意した．

4.3.3 流れ図課題実施後の自由記述問題

流れ図課題実施後に，プログラムと流れ図の，拡張前後の変化に関する自由記述課題を行う．自由記述課題は 2 問ある．表 2 にその具体的な内容を示す．

実験群では，OOP の利便性を非 OOP との比較によって実感させるという目的がある．そのため，問 1 では，OOP と手続き型のデータ構造の差を確認させ，問 2 では，その差が両手法のプログラミングの手間に及ぼした影響について考えさせる．この 2 つの問の狙いは，データ構造自体は両手法とも同一であるが，OOP では OOP の基礎概念が作用している分，分岐処理の追加等の拡張のしやすさが変わることに気づかせることにある．統制群では，問 1 で OOP のみの拡張の前後でのデータ構造の変化の有無を指摘させる．問 2 ではプログラミングやフローチャートの拡張作業を行って実感したことについて聞き，実験群と比べて回答にどのような差がでるかを調査する．

表 2 流れ図課題実施後に行う自由記述問題

	実験群	統制群
問１	OOP と手続き型で，フローチャートにどのような差が出ましたか？	プログラム拡張や，フローチャート拡張を行ってみて，プログラム内ではそのデータ構造にどのような変化が起きましたか？具体的に述べてください．
問２	また，その差によって OOP と手続き型でプログラミングのしやすさに影響は出ましたか．出たとすれば，どのような影響ですか．	OOP のプログラム拡張や，フローチャート拡張を行ってみて感じたことを教えてください．

4.3.4 事後問題

今回は，全てのプログラミング課題と流れ図課題が終了したあとに OOP の基礎概念や利便性に関する自由記述課題（事後問題）を実施した．この事後問題は，ID の第一原理[7]に則り，利便性と基礎概念の関係性を課題の例示を使って振り返らせたあと，その知識を一般化（統合）させて説明させるという手順を踏んでい

る．具体的な内容を一部，表 3 に示す．問 1,2 では，継承とオーバライドがもたらす作用と利便性について，具体的なプログラミング課題の内容を例に取りながら，OOP の基礎概念と利便性の結びつきに関して回答してもらう．問 6～8 では，問 1～5 を受けて，OOP の基礎概念と利便性の結びつきに関して一般論を述べてもらう．実験群では比較学習を行っているため，比較学習に関する問いを追加してある．

表 3 事後問題

問題番号	問題内容
1	OOP の各課題で作成した各サブクラス内で，一部の変数が宣言せずに利用できたのはなぜでしょうか？その理由を説明してください．
2	(1)OOP の各課題で様々なクラスを追加していく際に，小計メソッド shoukei や情報表示メソッド hyouji を再定義せずそのまま使いまわせたのはなぜでしょうか．プログラム内で作用した概念などの名前を挙げて具体的に説明してください． (2)また，そのようなことが実現することで得られるメリットは何でしょうか？
6	オブジェクト指向の利便性の 1 つである拡張性（要素の追加のしやすさ）は，プログラムの規模が大きくなる程得られます．それは，手続き型と比べてどのような部分が違うからでしょうか。
7	オブジェクト指向のもう一つの利便性である保守性は、オブジェクト指向の概念によるバグの入りにくさやデバッグのしやすさによって得られます。それは，デバッグのしやすさが、手続き型とオブジェクト指向でどのように違うからでしょうか
8	このような利便性は，OOP のどのような概念よって実現されますか．概念の名称をすべて列挙してください．

4.4 学習の組み立て

前回は，クラスや継承，多態性といった基礎概念を課題ごとに個別に意識させるような学習設計であったが，基礎概念と利便性を結びつけて考えさせることが困難であった．そこで今回の実験では，OOP の基礎概念を 1 つずつ理解させるのではなく，3 段階の課題の中で，OOP の基礎概念と利便性の結びつきを体感させていくことを意図した．

図 3 に今回の実験の流れを示す．1～2-2 では，プログラムの拡張内容はそれぞれ異なるが，実施の流れや目的はすべて同じである．プログラミング拡張を実施した後，流れ図拡張（描き込み＋自由記述）を実施する．流れ図拡張では，拡張箇所を描きこむことでプログラムのデータ構造を確認させる．その後は，プログラミング拡張と流れ図の変化についての問を設け，自由記述で回答させる．図 2 は，課題 1，2-1，2-2 を終え，拡張が完了した OOP のプログラムのクラス図である．

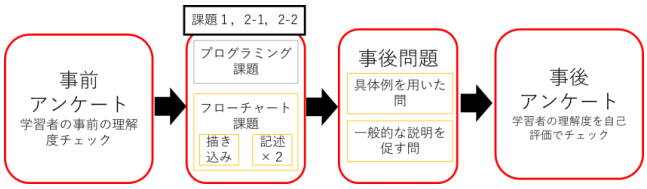

図 3 今回の実験の流れ

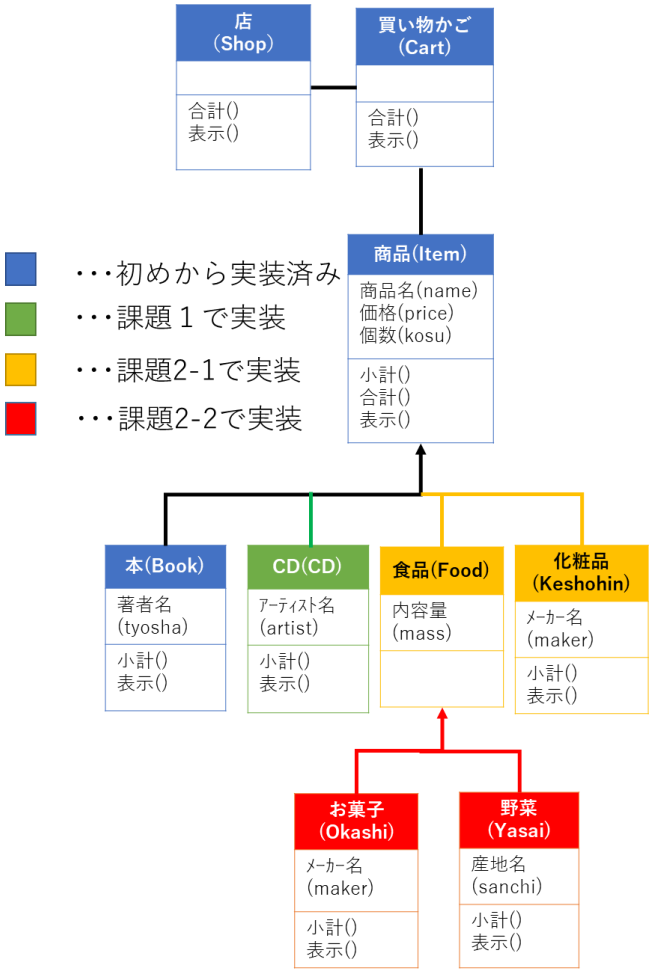

図 3 機能拡張した OOP のクラス図

5. 実験結果

5.1 実験概要

対象とした学生は，情報系学科 3 年生 7 名，4 年生

3 名の計 10 名である．学習者の条件は，手続型プログラミングの経験があり OOP の経験があるが概念を詳しく理解していない学生とした．この 10 名を実験群・統制群各 5 名に分けた．2 年次の java の演習講義の成績を基準に，両群でプログラミング能力が均等になるようにした．実験群では，OOP と手続き型の比較学習を行わせる．統制群では，OOP に関する課題のみを与える．比較学習の有無が，どれだけ学習者の OOP への理解度向上に繋がるかを確かめる．実験期間は 1 週間とし，その中で他人と相談せずに各自で課題に取り組んでもらった．使用言語は Java で，開発環境は eclipse である．

5.2 プログラミング課題の評価

表 4 に，プログラムソースと流れ図拡張の問題に対する各群の正答者数を示す．課題 1，2-1 に関しては，実験群，統制群ともに全体的に正答率がよく，プログラム拡張後の実行結果やデータ構造に大きな乱れは見られなかった．一方で課題 2-2 の「ソースの差異」の学習項目に関して，実験群の学習者 C，D が継承に関する，共通の誤りを起こしていた．ソースの差異の項目では，学習者が拡張した OOP，手続き型のプログラムのデータ構造にズレが生じていないかどうかを確認する．課題 2-2 の OOP の拡張では，Food クラスを継承した Okashi クラス，Yasai クラスを新たに作成する．学習者 C，D はこの 2 つのサブクラスを Food クラスを継承せず，直接 Item クラスに継承させていたため，図 3 で示したようなデータ構造になっていない．また，手続き型の拡張に関しても OOP の誤答箇所と対応する箇所に誤りが見られた．手続き型では，お菓子・野菜の処理は食品の小計計算のメソッド shoukei_food メソッドの内容として呼び出すことが理想である．しかし，学習者 C，D の手続き型の回答では，お菓子・野菜に関する処理を，元々定義していた shoukei_food メソッドを呼び出さず，新規にメソッドを作成していた．そのため，OOP のプログラムと同様にデータ構造の誤りを起こしてしまったと考えられる．この継承の適用箇所に関する誤りは，前回の実験に参加した一部の学習者にも見られている．

表 4 プログラムソースと流れ図拡張の問題に対する各群の正答者数（各群 5 名中）

評価項目＼学習課題	課題 1		課題 2-1		課題 2-2	
ﾌﾟﾛｸﾞﾗﾑの実行結果（OOP）	実	統	実	統	実	統
ﾌﾟﾛｸﾞﾗﾑの実行結果（手続き型）	4	5	5	4	4	5
ソースの差異（OOP）	4	－	5	－	5	－
ソースの差異（手続き型）	4	5	3	4	3	5

5.3 流れ図課題の評価

表 5 に流れ図描き込み課題に対する各群の正答者数を，表 6 に，実験群の学習者 A の流れ図課題の記述部分の回答例を示す．

まず，流れ図描き込み課題の評価について説明する．課題 1，2-1 については両群ともに正答率が高い．しかし，課題 2-2 は若干正答率が低い．プログラミング課題と同様，継承による処理に関する誤りが見られた．プログラミング課題で Okashi クラス，Yasai クラスが Food クラスを継承せず，Item クラスで継承する図を作成した学習者が数名見られた．

次に流れ図課題の記述部分に関する評価について説明する．学習者 A は，OOP と非 OOP の違いがデータの分岐処理にあることに気づき，その違いによってプログラミングの手間が減るという点を指摘した．実験群の学習者は，プログラミング課題だけでなく，流れ図課題でも非 OOP との比較学習が行えるため，OOP と非 OOP とのデータ構造や処理の差が視覚的に確認できる．そのため，実験群の学習者は，「OOP では if 分岐の代わりにクラスを利用し，メソッドを再定義せずに利用できることが両手法間での違いである」ことに気づく傾向にあった．このことが，「多態性が拡張性や保守性といった OOP の利便性と関係性がある」と気づくきっかけになったのではないかと考えている．

表 5 流れ図描き込み課題の正答者数（各群 5 名中）

評価項目＼学習課題	課題 1		課題 2-1		課題 2-2	
流れ図描き込み（OOP）	5	－	4	－	3	－
流れ図描き込み（手続き型）	5	4	5	4	3	2

表 6 流れ図課題（自由記述）の回答例

学習者	1）OOP と手続き型で，フローチャートにどのような差が出ましたか？
実験群/A	小計メソッドの呼び出しや製造元の名前を出力する際の品種の識別が、手続き型の場合だと if 文によって行われているが、OOP の

	場合だとクラスによって行われていることが違う。
	2)また,その差によってOOPと手続き型でプログラミングのしやすさに影響は出ましたか.出たとすれば,どのような影響ですか.
A	手続き型は処理ごとに品種による場合分けをしなくてはいけないが、OOP では品種のクラスごとに処理内容を決めることができるため、OOP のほうがプログラムがしやすいと感じた。

5.4 事後問題の評価

事後問題の回答内容を，表 1 で示した基準で評価した結果を表 7 に示す．レベル 2 の「OOP と利便性と基礎概念の関係性を，具体例を使いながら説明ができる」程度の理解度に達した学習者が多い．

次に，表 8 に事後問題の回答例を示す．まずはじめに，具体例を用いた OOP の利便性に関する回答例として,実験群の学習者 A と統制群の学習者 F の問 2 の回答例をそれぞれ抜粋して説明する．問 2 は，継承とオーバライドを組み合わせて起こる作用と，それによる利便性について，課題を例に説明する問題である．各商品クラスに各メソッドが同一の名称でかつ再定義なしで利用できた理由と，それが引き起こすメリットに関して正しく述べられていれば正解とする．学習者 A，F は両者とも，問 2 の(1)でオーバライドによってメソッドの再定義を防ぐことを指摘し，(2)でそれによって修正の手間が省けるといった趣旨の回答が見られたため，正解とした．他の両群の学習者の回答も，同様の意味合いで読み取れる内容が多かったため，全体的な正答率が高い傾向にあった．

次に,OOP の利便性に関して一般的な説明を促す問の回答例として，学習者 A,F の事後問題の問 6 の回答例について説明する．問 6 は OOP の利便性の 1 つである拡張性が生じる理由を問う問題である．表 6 に示した統制群の学習者 F の回答は，拡張性が実現した理由に継承とオーバライド (=多態性) が関与しており，これらの概念がどのように作用したかということに関してある程度正確に述べることができている．一方で実験群の学習者 A は，拡張性が得られる事例を述べているが，それが得られる理由を基礎概念と結びつけて説明しておらず,説明が抽象的である.このため,OOP の利便性と基礎概念の結びつきに関して，本質的に理解できているかどうかの判断ができなかった．全体的な傾向としては，学習者 A のような回答をしている学習者が多かった．

表 7 学習者のレベル到達度

レベル	実験群	統制群
0	-	-
1	-	I
2	A,B,C,D	G,H
3	E	F,J
4	-	-

表 8 事後問題の回答例（問 2，6）

問 2(1)　OOP の各課題で様々なクラスを追加していく際に,小計メソッド shoukei や情報表示メソッド hyouji を再定義せずそのまま使いまわせたのはなぜでしょうか.プログラム内で作用した概念などの名前を挙げて具体的に説明してください.
（学習者 A）**スーパークラス(Item クラス)のメソッドをサブクラス(Book クラスや CD クラス等)がオーバーライドしているため**
（学習者 F）**サブクラスはスーパークラスで定義されているメソッドを受け継ぎ，そのメソッドを　オーバーライド(上書き)して宣言することができるため，再定義する必要がないから.**
(2)　また,そのようなことが実現することで得られるメリットは何でしょうか？
（学習者 A）サブクラスごとに小計の求め方を変更したいといったような細かい仕様変更の際、**オーバーライドを利用することによって、修正する箇所を少なくすることができること。**
（学習者 F）メソッドを新しく定義する必要がなくなることで，**バグの発生する範囲を狭めることが可能となる.**
問 6 : オブジェクト指向の利便性の 1 つである拡張性は,プログラムの規模が大きくなる程得られます．それはなぜでしょうか。
（学習者 A）手続き型は処理ごとに品種による場合分けをしなくてはいけないが、OOP では品種のクラスごとに処理内容を決めることができるという部分が違うため
（学習者 F）**クラスを継承してサブクラスを創り出すことでオーバーライドを可能にし，違う操作であるが似たような処理を同一のメソッド名で操作できるようにしている**から.

5.5 事前・事後アンケート結果

表 9 は，実験前後にアンケートを実施し，OOP の各基礎概念や基礎概念と利便性の関係性について，学習者の理解度を自己評価させたものである．若干統制群の方が実験群よりも自己評価の伸びが高い．実験群では，オーバライドの項目に関して，実験後も 3 未満の評価をした学習者が数名いた．基礎概念と利便性の結びつきについては，両群とも 4~5 という高評価を付ける学習者がほとんどであった．

表 9 事前・事後アンケート結果

学習者 / アンケート項目	実験群平均		統制群平均	
	事前	事後	事前	事後
クラス	3	3.2	4	4
インスタンス	3	3	3	4
メソッド	3	3.2	3	4
継承	2.8	3	3	4
スーパークラス・サブクラス	2.8	3	3	4
オーバーライド	2.6	2.8	3	4
多態性	2.4	3	2	3
基礎概念と利便性の関係性	-	4	-	4

表 10 事後問題の回答例（問 8）

学習者	問 8）OOP の利便性は，OOP のどのような概念によって実現されますか. 概念の名称をすべて列挙してください
実験群/A	オブジェクト・**クラス**・インスタンス・カプセル化・**多態性**
B	**クラス**・インスタンス・コンストラクタ・**継承**・**多態性**・カプセル化
C	**クラス**・インスタンス・**継承**・カプセル化
D	**クラス**・インスタンス・オブジェクト・カプセル化
E	**クラス**・**継承**・カプセル化・**ﾎﾟﾘﾓｰﾌｨｽﾞﾑ**
統制群/F	**クラス**・**継承**・オーバーライド・**多態性**
G	**継承**・オーバーライド
H	概念モデリング・ﾕｰｽｹｰｽ・ﾎｯﾄｽﾎﾟｯﾄ
I	extends（継承の宣言）
J	**クラス**・メソッド

6. 考察と今後の課題

本研究では，OOP と非 OOP の比較学習を支援する改善策を提案し，学習者の OOP の利便性と基礎概念の結びつきに関する理解度の向上を狙った．この章では，実験結果を踏まえた考察を行う．

6.1 実験群と統制群の理解度の共通点と相違点

実験群と統制群の OOP の理解度に関する共通点について説明する．事後問題の回答内容から，主に利便性の特徴については,「拡張の際の手間が省ける」,「修正が楽になる」といった形で両群ともに多くの学習者が理解できていた．しかし，利便性がなぜ生じるかという点について, OOP の基礎概念の作用と結びつけて説明できていた学習者は両群ともに少なかった.

相違点としては, 実験群の方が統制群よりも OOP の利便性と基礎概念の結びつきについて意識している傾向があったことが分かった. 表 10 に, 両群の学習者全員の事後問題問 8 の回答例を示す. 問 8 は 2 つの利便性を実現する OOP の基礎概念の名称を答えさせる問である. その中で, OOP の利便性を実現することに直結する「多態性」の概念を含んでいた学習者が，実験群に 3 人，統制群に 1 人いた．このような結果になった要因として，実験群がプログラミング課題に加え流れ図課題を併用し，そこで非 OOP との比較を行なったことで，このような傾向が見られたのではないかと考える.

6.2 OOP の利便性と基礎概念の結びつきの理解度

OOP の利便性と基礎概念の結びつきの理解度に関しては，4 章で述べた総合評価から，実験群・統制群ともに「プログラム内の処理などの具体例を使いながらであれば，利便性と基礎概念の関係性について簡単に説明ができる」レベルに達した学習者が多かった．しかし,「一般的な説明として，OOP の利便性の特徴とそれが生じる理由を説明できる」レベルまで理解度を引き上げることは困難であった．今回の改善策に関する考察を以下に示す.

6.2.1 評価基準の改良に関する効果

本研究では，学習者の OOP に対する理解度向上の基準を,「OOP の利便性と基礎概念の結びつきに関して正しく説明できる」ことに合わせている．学習者の理解度のレベルを上記のような基準をもって細分化することで，ある程度の正確さをもって学習者の理解度を判断するようにした．一部の学習者の回答には，設定したレベルの条件を一部しか満たしていないものなど，レベルに当てはめることが困難な事例がいくつかあった．今後は，学習者の回答に対する分析をより正確に行うため，今回設定した評価基準をベースにさらにレベルを細分化するなどの対策を行う

6.2.2 流れ図課題に関する効果

実験群では, 非 OOP と OOP の比較を行いながら流れ図課題に取り組んでもらうことで, OOP のプログラム内で行われている処理を捉えやすくさせた．これによって, OOP の利便性に気づかせ，それがどの基礎概念によって生じているのかという点を統制群より意識させやすくすることを狙った.

実験の結果，全体的に実験群と統制群で回答内容に大きな差はなく，両群の学習者とも記述問題では，プ

ログラムのデータ構造や拡張した際の手間などに関して正しく指摘していた学習者が多かった．実験群の一部の学習者については，OOP と非 OOP でデータ分岐の処理に違いがあるという点を，課題の処理の例に基づいて気づかせることができた．統制群の学習者は，非 OOP との比較なしで拡張前後のプログラムの変化について気づきを得る傾向にあった．両群でこのような結果が出た理由に，プログラミング課題実施直後に流れ図拡張と利便性に関する記述問題に取り組むことで，プログラム拡張の前後の変化を掴みやすくなったことが考えられる．実験群・統制群の総合評価でレベル２の学習者が全体的に多かったのは，プログラミングに加え流れ図課題を併用することで具体例を使って利便性と基礎概念を意識させたことが要因であると考えられる．しかし，一方で「利便性がどのように生じるか」という点については問を設けておらず，あくまで「プログラムにどのような変化があったか」,「それによってプログラムにどんな影響が生じたか」という点でしか考察をさせていなかった．この点が，事後問題の回答内容にも影響を及ぼした可能性がある．したがって，現時点では，プログラム内で利便性が生じる理由を問う問題を入れるなどの対策が考えられる．

6.2.3 事後問題の内容・配置の変更に関する効果

事後問題については，前回から内容と配置の改善を行うことで学習者の OOP に対する理解度定着を狙った．問 1~5 はプログラミング課題の拡張の例を示しながら OOP の利便性と基礎概念の関係性について考えさせることを目的としている．これらの正答率は実験群・統制群ともに高かった．問 6～8 では，問 1～5 を踏まえて具体例を使わずに OOP の利便性に関する一般的な説明を促すことを目的としている．この 3 問については正答率にばらつきがあり，ほとんどの学習者は利便性の一般的な特徴を述べるのみで，それがどの基礎概念によってどのように実現されるかについて説明できた者は少なかった．この原因として，今まで具体例を使って考えていたものに対し，急に一般的な問に切り替えて取り組ませてしまったことが挙げられる．そのため，学習者の OOP の利便性と基礎概念に関する知識が不完全なままアウトプットさせてしまっていたと考えられる．

6.3 今後の方向性

今後の課題として，今回提案したアプローチのブラッシュアップと転移課題の追加が考えられる．前者については，例えば評価基準のレベルの細分化や流れ図や事後問題の問いかけの仕方を改善することが挙げられる．後者に関しては，既に作成したサンプルプログラム以外のプログラミング課題を事後問題実施前に取り組ませることが挙げられる．学習者の OOP に関する知識を転移課題で応用することによって，OOP の利便性と基礎概念の結びつきに関する説明が一般的にできるレベルに近づかせることが可能なのではないかと考える．

参 考 文 献

(1) 立山秀利，:“Java のオブジェクト指向がゼッタイにわかる本”，秀和システム (2016)

(2) 中鉢直宏，伊藤一成：“オブジェクト指向プログラミング教育における LEGO を用いた体験型課題の試み，情報処理学会研究報告，Vol.2014-CE-124，No.8，pp.1-6 (2014)

(3) 大城正典，永井保夫：“プログラミング初学者を対象としたオブジェクト指向プログラミング教育システムの提案　―オブジェクト指向の基本概念の理解に基づいたプログラムの作成・実行支援機能を中心として―”，情報教育シンポジウム 2016 論文集，pp.114-121 (2016)

(4) 三浦元喜，杉原太郎：“オブジェクト指向言語におけるクラス定義の意味とオブジェクトの振舞いを理解するためのワークベンチ”, 情報教育シンポジウム, pp.43-49 (2011)

(5) 石川裕季子，松澤芳昭，酒井三四郎：”オブジェクト指向言語におけるポリモーフィズムの概念を理解するためのワークベンチ”，教育システム情報学会誌，Vol.31，No.2，pp.208-213 (2014)

(6) 竹川夏実, 仲林清：”オブジェクト指向プログラミングの利便性に着目した学習手法”， 教育システム情報学会研究報告，Vol.32，No.2，pp21-28 (2017)

(7) C．M．ライゲールスマン，A．A．カー＝シェルマン：“インストラクショナルデザインの理論とモデル　共通知識基盤の構築に向けて”，北大路書房，pp.58-59 (2016)